International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:1, 2015

Adapting the Chemical Reaction Optimization Algorithm to the Printed Circuit Board Drilling Problem

Taisir Eldos, Aws Kanan, Waleed Nazih, Ahmad Khatatbih

***Abstract*—**Chemical Reaction Optimization (CRO) is an optimization metaheuristic inspired by the nature of chemical reactions as a natural process of transforming the substances from unstable to stable states. Starting with some unstable molecules with excessive energy, a sequence of interactions takes the set to a state of minimum energy. Researchers reported successful application of the algorithm in solving some engineering problems, like the quadratic assignment problem, with superior performance when compared with other optimization algorithms. We adapted this optimization algorithm to the Printed Circuit Board Drilling Problem (PCBDP) towards reducing the drilling time and hence improving the PCB manufacturing throughput. Although the PCBDP can be viewed as instance of the popular Traveling Salesman Problem (TSP), it has some characteristics that would require special attention to the transactions that explore the solution landscape. Experimental test results using the standard CROToolBox are not promising for practically sized problems, while it could find optimal solutions for artificial problems and small benchmarks as a proof of concept.

***Keywords*—**Evolutionary Algorithms, Chemical Reaction Optimization, Traveling Salesman, Board Drilling.

I. Introduction

MODERN life heavily relies on optimization as a keystone in science and engineering when applied to the business and industry problems; most of the problems can be formulated as optimization problems; like scheduling, cell placement, stock market trend prediction, etc. Algorithms for finding optimal solutions for practically sized problems in reasonable time do not exist and may not even in the future, but if the goal is relaxed to near optimal then a wide range of approximate algorithms do exist. For example, the class known as nature-inspired optimization techniques are typically general-purpose population-based techniques, also known as Evolutionary Algorithms (EA), build on the natural biological processes. Although EA refers to the biological process, it has been used with any process of iteratively changing a group of possible solutions towards the best possible one for the problem in hand. Amongst the most popular ones are: Genetic Algorithm (GA) [1], Particle Swarm Optimization (PSO) [2], Memetic Algorithm (MA) [3], [4], Differential Evolution (DE) [5], Ant Colony Optimization (ACO) [6], and Harmony Search (HS) [7]. Many of them are inspired by the biological process, varying in scale from the genetic structure to the living things behavior, and recently the CRO, which was proposed by [8]. The CRO is inspired by the nature of chemical reactions, and has been applied to solve many problems successfully, outperforming many evolutionary algorithms. Gravitational Search Algorithm (GSA) [9], [10] is an example of evolutionary algorithms that build the physical phenomenon to evolve a set of solutions towards a near optimal one. Another example is the Simulated Annealing (SA) [11], which builds on the annealing process in which the physical properties of iron depends on the cooling schedule after melting. This algorithm has a hill climbing feature but differs from the CRO in that it works on one solution rather than a population. The CRO behavior becomes close to the SA under certain conditions.

The standard CRO uses different elementary actions on a set of molecules to reach an equilibrium or minimal energy state. The actions consist of two ineffective collisions, on-wall and inter-molecular, to achieve intensification (or exploitation in terms of solution space), and two others; decomposition and synthesis, to achieve diversification (or exploration in terms of solution space). In our implementation, we will use similar actions but with names that are more consistent with algorithmic significance in the search process.

During the search, and based on the parameters settings, the CRO may demonstrate behavior of both SA and GA, and hence may have a potential to tackle problems which have not been successfully solved by other metaheuristic. CRO has not been heavily explored; some problems were solved successfully due to some reports, but we don't know yet which classes of problems are suitable for CRO and which are not. There is no easy answer at this moment, and like many other evolutionary algorithms, it takes a while to figure out as the research community applies this method to more and more problems.

II. Related Work

Chemical Reaction Optimization (CRO) [8] is a recently proposed general-purpose optimization metaheuristic. It mimics the interactions of molecules in chemical reactions, driving towards the lowest state of free energy. CRO has demonstrated its ability to solve many real-world optimization problems. In [12], the authors employ CRO to solve an optimization problem called population transition problem. The objective is to maximize the probability of universal

Taisir Eldos is with the Jordan University of Science and Technology, Ibid - Jordan, currently in a sabbatical leave at Salman bin Abdulaziz University, Alkharj - Saudi Arabia (e-mail: eldos@sau.edu.sa.).

Aws Kanan, Walid Nazih, and Ahmad Khatatbih are with the Salman bin Abdulaziz University, Alkharj - Saudi Arabia (e-mail: a.kanan@sau.edu.sa, w.nazih@sau.edu.sa, a.khatatih@sau.edu.sa).

streaming by manipulating population transition probability matrix. Simulation results show that CRO outperforms many commonly used strategies for controlling population transition in many practical P2P live streaming systems. Reference [13] develops a CRO-based cognitive radio channel allocation algorithm, it studies three utility functions for utilization and fairness, with the consideration of the hardware constraint. The proposed algorithm always outperforms the others dramatically. Reference [14] proposes a real-coded version of CRO called RCCRO to solve continuous optimization problems. The performance of RCCRO is compared with a large number of optimization algorithms on a large set of standard continuous benchmarks.

The PCB drilling problem is an instance of the Traveling Sales Problem (TSP), a widely studied combinatorial optimization problem, which is concerned with finding the shortest tour that a salesman has to visit through all the cities. It has many applications in different engineering and optimization problems [15]. Several metaheuristic approaches have been developed to solve the TSP problem such as Simulated Annealing [16], Tabu Search [17], Genetic Algorithms [18], Variable Neighborhood Search [19], Neural Networks [20], Ant Colony Optimization [21], and Particle Swarm Optimization [22]. In [23], TSP is used as a case study to show the capabilities of the Ant Colony Optimization (ACO) algorithm to find the best solution in terms of the shortest tour length. The paper presents experimental results on a benchmark data to show how it could improve ACS algorithm. Reference [24] develops a new fuzzy-logic based ACO algorithm, considering the uncertainties that can be found in the pheromone and the heuristic factors. The proposed technique enables the artificial ant to choose the best oncoming step based on the values of the probabilities and their corresponding fuzzy levels. It produces an optimal solution in the form of an optimal value and its corresponding fuzzy level.

III. Chemical Reaction Optimization

The CRO algorithm starts with a number of molecules picked at random or through a mechanism for locating feasible starting points in the landscape. Few elementary operations are applied to those molecules iteratively producing more fit ones until some stopping criteria is met. In the main reactions, the ones meant to exploit subspaces, the number of inputs and outputs are equal and hence do not change the population size no matter how often they are called. However, the other two reactions, the ones meant to explore the space, are either population increasing or population decreasing, and unless they have the same number of occurrences, the population size will tend to increase or decrease. Fortunately, if the population decreases, the algorithm will tend to apply decomposition to avoid the single solution population. However, both extremely small and extremely large populations are undesirable, as the first is less effective in search while the second is a computational burden.

Based on the number of molecules involved and the number of molecules produced, we divide the CRO elementary reactions into four classes:

- 1-to-1 processes, one molecule is involved to produce one molecule; a process in which a molecule structure is deformed through a minor or a major structural change, we call this process deformation.
- 1-to-2 processes, one molecule is involved to produce more than one molecule; typically two molecules are derived from one, we call this process decomposition.
- 2-to-1 processes, two or more molecules are involved to produce one molecule; a process in which the properties of many are expressed in one, we call this process combination.
- 2-to-2 processes, two molecules are involved to produce two molecules; a process in which properties of both are transferred into two new molecules; we call this process collaboration

In the context of developing new solutions from a set of ones in hand at a certain time, we will call the first two actions, the ones that involve one molecule, D-type (Deformation, Decomposition), while we call the other two, the ones that involve 2 molecules, C-type (Combination, Collaboration). Related literature uses chemical reactions terms like synthesis instead of combination, and intermolecular collision instead of collaboration, but we feel better off using terms that express the processes behavior as they take place in the search process. Starting with a set of molecules representing valid solutions to the problem in hand, those processes are meant to generate better sets over time, until we reach an optimal solution; convergence. It is quite important for convergence to carry out operations that provide both exploration and exploitation. The relative effect of each type on the exploitation and exploration is shown in Table I, where ** means primary and * means secondary, based on the contribution to convergence. In fact, no one action is solely responsible for either exploration or exploitation.

TABLE I
Actions and Significance

Action	Cardinality	Exploration	Exploitation
Deformation	1-to-1	*	**
Decomposition	1-to-2	**	*
Combination	2-to-1	**	*
Collaboration	2-to-2	*	**

The CRO algorithm is simple, we initialize a set of parameters based on the instance to solve, and we repeat until the stopping criterion is satisfied. In each loop we go for C-type or D-type path based on the present value repressing the C-type actions rate. And in each path, we check for decomposition or synthesis condition if met, or else we perform deformation or collaboration instead. The pseudocode below shows an outline of the algorithm, along with basic definitions of the terms used.

1. Set The CRO Initial Parameters

```
2.  Pick Initial Set of Molecules
3.  Pick a random number p ϵ [0,1]
4.  If p < CAR    {
                        Pick a molecule at random
                        If CCS      { // Apply Combination
                                    }
                        else
                                    { // Apply Collaboration
                                    }
                        }
        else
                        {
                        Pick two molecules at random
                        If CDS      { // Apply Decomposition
                                    }
                        else
                                    { // Apply Deformation
                                    }
                        }
        endif
5.  Keep Current Optimal Molecule
6.  If NOT SCS GOTO 3
7.  Report Optimal Molecule
```

CAR means C-type Actions Rate
CCS means Combination Criteria Satisfaction is true
DCS means Decomposition Criteria Satisfaction is true
SCS means Stopping Criteria Satisfaction is true

The stopping criteria used is a preset number of repeats with a preset number of iterations for each. The values CAR, CCS and DCS were varied based on the test to be carried out.

IV. PCB Drilling Problem

Printed Circuits Board (PCB) manufacturing is a major component of computers and electronic equipments in general. The number of holes of various diameters varies from few tens to few hundreds or even thousands. The time to drill the holes depends on the order by which the numerically controlled drilling machine drill the holes. This problem can be viewed as an instance of the well known Traveling Salesman problem (TSP) with a time matrix instead of distance matrix.

The Printed Circuit Board Drilling problem can be viewed as an instance the Traveling Salesman Problem, where the cities map to holes and the distance matrix maps to the time matrix, which represents the time to fly the drill bit between two holes. To make use of the CROToolBox, we represent each candidate solution with a string whose length is the problem size, i.e. number of holes to order for drilling. While processing those solutions as molecules, we apply a set of actions towards getting more fit ones taking into account that none of those actions results in an invalid solution; every hole number has to appear exactly once in the string. The actions may produce solutions that look different while they are in reality the same from the fitness point of view, for example, the two strings:

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 – 9 and 3 - 4 - 5 - 6 - 7 - 8 - 9 - 0 - 1 - 2

represent two solutions while they have the same fitness and they represent the same order. Such strings may show up in the population over time and no attention will be paid to avoid them, as they represent different schemata in fact, and applying the same action to each would result in two different outcomes. For example, if the deformations action is to operate on the 4th entry, then we can get:

0 - 1 - 2 - 4 - 3 - 5 - 6 - 7 - 8 – 9 and 3 - 4 - 5 - 7 - 6 - 8 - 9 - 0 - 1 - 2

which are not the same. Hence, we believe that this would avoid computational burden and results in better exploration of the solution space.

D-Type and C-Type Actions

Two types of actions are used in the CRO search; D-type and C-type. The D-type actions use one molecule to generate either one or two molecules. Deformation is a process in which a molecule is slightly perturbed to generate a new molecule, while Decomposition is a process that generates two molecules out of one, passing to each one part of its structure. The C-type actions use two molecules to generate one or two molecules. Combination is a process in which two molecules combine or unite to generate one, carrying part of each structure, while Collaboration is a process in which two molecules have some sort of crossover to generate two ones with some degree of similarity; each one have parts of the two molecules involved. The following section explains how to carry out those actions using a small instance of 10 holes, showing the input(s) and outputs(s) of each action.

1-to-1 Actions (Deformation)

This process injects some structural change, so the molecule gets deformed. These deformations can be minor; swapping a randomly selected entry with its neighbor, major; swapping two randomly selected entries, or hard; by selecting three entries at random and swapping the first with the second, the second with the third, and the third with the first. For example:

Input
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9
minor; one point
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 – 9
Output
0 - 1 - 2 - 3 - 5 - 4 - 6 - 7 - 8 – 9
major; two points
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 – 9
Output
0 - 1 - 2 - 3 - 7 - 5 - 6 - 4 - 8 - 9
hard; three points
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 – 9
Output
0 - 7 - 2 - 3 - 1 - 5 - 6 - 4 - 8 - 9

1-to-2 Actions (Decomposition)

This process decomposes a molecule into two new ones to enrich the population in its solution space representation; one

way to carry this out is by picking a random point, and generating two molecules by keeping lower or upper part and randomizing the rest. For example:

Input
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9
Outputs
0 - 1 - 2 - 3 - 4 - 7 - 9 - 6 - 8 – 5 and 2 - 3 - 1 - 0 - 4 - 5 - 6 - 7 - 8 - 9

2-to-1 Actions (Combination)

This process combines two molecules into one; it constructs a molecule whose entries are alternatively form the first and the second without duplication. For example:

Inputs
0 - 6 - 1 - 3 - 4 - 8 - 2 - 7 - 5 – 9 and 7 - 2 - 8 - 5 - 6 - 0 - 3 - 9 - 4 - 1
Output
0 - 7 - 6 - 2 - 1 - 8 - 3 - 5 - 4 - 9

2-to-2 Actions (Collaboration)

This process is possibly the most significant due to its major contribution to exploitation; we pick two molecules and mate them to produce new two molecules carrying the properties of both. This can be carried out by picking two random molecules and make a cut in each; keep the first part and fill the other from the other molecule. For example:

Input
0 - 6 - 1 - 3 - 4 - 8 - 2 - 7 - 5 – 9 and 0 - 7 - 2 - 8 - 5 - 6 - 1 - 3 - 9 – 4
Outputs
0 - 6 - 1 - 3 - 4 - 7 - 2 - 8 - 5 – 9 and 0 - 7 - 2 - 8 - 6 - 1 - 3 - 4 - 5 - 9

In all cases, the number of molecules has to be kept within reasonable limits to match the computational resources. Unless the decomposition and synthesis rates are equal, the population will tend to inquires or decrease by time. A way around that is to drop molecules if a list is reached, like the least fit ones, and to split one to two for example when it goes quite low. It is like thinking of a larger container holding the container in which reactions take place, if the inner container wall is pressure sensitive such that when pressure inside is low due to smaller population then more molecules from the outer container go in, and if it the pressure is high due to larger population then some leak out.

V. Experiments

First set of tests were conducted using the CRO Tool Box with the default set of parameters. Small problems have shown good results and the algorithm could even find an optimal solution, while for medium sized problems the final solution was an acceptable even with 20 repeats.

Table II shows 10 runs (repeats) for various number of iterations using a set of default parameters; Initial population Size: 150, Initial Kinetic Energy: 1000, Collision Rate: 0.2, Energy Loss Rate: 0.2, Decomposition Threshold: 1300, Synthesis Threshold: 10. In Table II, Error is computed as the difference between the optimal and the found one divided by the optimal.

TABLE II
Best Solution for 4 Problems

Problem		10,000		100,000		1,000,000	
Size	Optimal	Best	Error	Best	Error	Best	Error
22	74.07	75.92	0.07	74.45	0.18	74.07	0.006
48	33523	92250	1.92	56300	0.75	41411	0.30
96	508.00	1438	2.04	1106	1.34	1069	1.22
130	6110.00	35313	5.04	17055	1.91	16157	1.73

Testing the PCB442 (Optimal cost 50783) against number of iterations using default parameters; Initial population Size: 150, Initial Kinetic Energy: 1000, Collision Rate: 0.2, Energy Loss Rate: 0.2, Decomposition Threshold: 1300, Synthesis Threshold: 10.

Table III shows the PCB442 benchmark test result. Clearly, the number of iterations has an impact on the solution quality, but not justifiable at large number of iterations; Going from 10,000 to 10 folds reduced the error by 2.06 while going from 1,000,000 to 10 fold reduced the error by only 0.25. In all cases, the performance of this algorithm with its default parameters is far less than expected.

TABLE III
Statistics versus Number of Iterations

Iterations	Average	Best	Std. Dev.	Error
10,000	555964	548811	3727	9.94
100,000	552748	542960	4517	7.88
1,000,000	329682	313044	8631	5.49
10,000,000	317130	313473	2558	5.24
100,000,000	310670	306077	2437	5.11

Table IV shows the effect of Decomposition Threshold; 10 runs, each 1,000,000 iterations. Small increments of the starting point of 1300 have no impact on the error, that we made large increments of 25,000. The minimum error occurred when the threshold reached 100,000 and started to increase again. This parameter is problem dependent.

TABLE IV
Impact of Decomposition and Synthesis Rates

Threshold	Decompose	Synthesis	Best	Error
1300	452	600	322463	5.34
25,000	31	180	160442	2.15
50,000	15	164	149574	1.94
75,000	10	159	143506	1.82
100,000	7	156	136443	1.68
125,000	6	155	143212	1.82
150,000	5	153	143873	1.87

To evaluate the impact of the synthesis rate, we carried out many runs, 100,000 iterations each, with fixed decomposition threshold, and noticed only minor change in performance even

with large adjustments of collision rate, kinetic energy, energy loss rate. Error fluctuated in the range 1.67 to 1.77 in no pattern.

The progress, cost and population size over time, is tested using 22-hole and 130-hole problems for 10,000 and 100,000 runs respectively, and the population, cost and number of reactions of each type is shown. Figs. 1 and 2 depict the cost and population size over time (iteration) for the 22-hole and 130-hole problems respectively. The CRO performance on small instances was consistently excellent, as it could always find an optimal solution, while it starts to decline even with instances that are far less than the practically commercial size. With PCB442 benchmark, the cost has never come even close to the optimal one, even with tens of millions of iterations and many repeats.

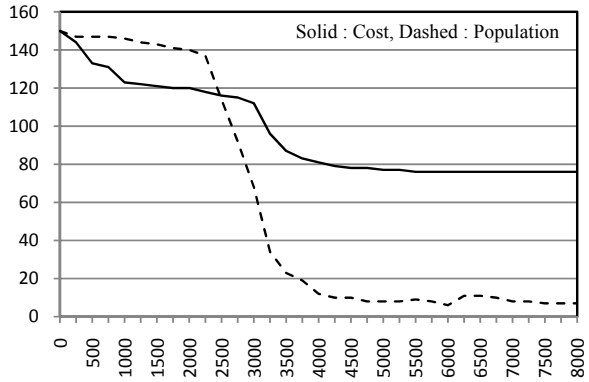

Fig. 1 Cost and Population over Time (22-hole instance)

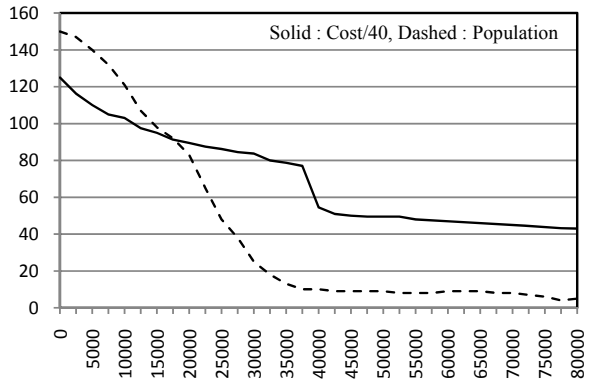

Fig. 2 Cost and Population over Time (130-hole instance)

Fig. 3 shows the reaction types involved over time. The test was conducted on a small instance and for 10,000 iterations.

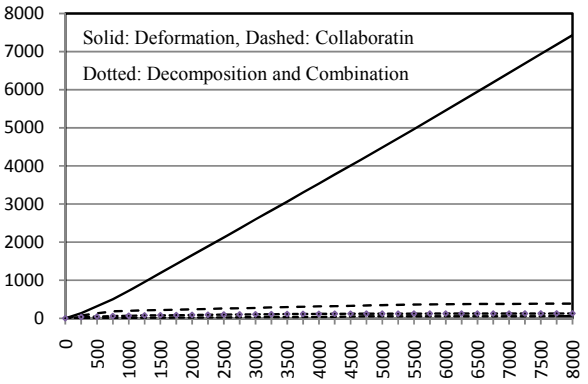

Fig. 3 Actions Count over Time

Clearly, decomposition and combination contribution is quite limited and almost have the same rate, while the collaboration is relatively small compared to the deformation. With this set of parameters the CRO behaves like SA algorithm. The CRO could find the optimal solution for the 22-hole problem many times when 10,000 iterations and default parameters were used, while the 130-hole problem solution was always far from optimal. However, it was noticed that major improvements in the objective function were achieved when the population size drops to small values.

VI. Conclusion

As a recent general-purpose optimization technique, the CRO builds on the nature of chemical reactions, mimicking the interactions of molecules in the form of elementary reactions. The sequence of the elementary reactions leads to exploring the solution space towards a minimum with a hope of a global minimum. The search parameters must be set to control the search scope like climbing hills and hence managing the energy is pivotal property of such an algorithm. Another advantage of this algorithm was its ability to adaptively change the population size and hence do more iterations in less time. Although the literature reports many successful applications of the CRO, we believe there is still much to do to realize if the PCB drilling problem, or the TSP as an umbrella, is fit or not for this technique, and thanks to the No- Free-Lunch Theorem [25] which tells us that all metaheuristic which search for extremes are exactly the same in performance when averaged over all possible objective functions, while one works well in a certain class of problems, it may not in another.

VII. Future Work

Due to the less than expected performance on benchmarks, we plan to modify the elementary reactions and possibly introduce new ones that enhance the effectiveness of the CRO algorithm for the PCB drilling problem in particular. One possible way to go is to consider the peculiar relations among sets of holes; like those representing two or four sides of a chip. This might require partitioning the holes in a way that

avoids randomized change to relatively stable connectivity patterns.

ACKNOWLEDGMENT

This work was funded by the Deanship of Scientific Research at Salman bin Abdulaziz University, Alkharj - Saudi Arabia, under the research project No. 94/T/33, during the main author's sabbatical leave from Jordan University of Science and Technology in the year 2012/13.

REFERENCES

[1] Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA, USA.
[2] Kennedy J, Eberhart RC (2001) Swarm Intelligence. MorganKaufmann, San Francisco.
[3] Ong YS, Lim MH, Chen XS (2010) Research Frontier: Memetic Computation Past, Present and Future. IEEE Computational Intelligence Magazine 5(2):24–36.
[4] ChenXS, OngYS, LimMH, TanKC (2011) A Multifacet Survey on Memetic Computation. IEEE Trans Evolutionary Computation 15(5):591–607.
[5] Price K, Storn R, Lampinen J (2005) Differential Evolution: APractical Approach to Global Optimization. Springer, Berlin.
[6] Dorigo M, Stutzle T (2004) Ant Colony Optimization. The MITPress, Cambridge, MA, USA.
[7] Geem ZW, Kim JH, Loganathan GV (2001) A New Heuristic Optimization Algorithm: Harmony Search. Simulation 76(2):60–68.
[8] Lam AYS, Li VOK (2010) Chemical-Reaction-Inspired Metaheuristic for Optimization. IEEE Trans Evolutionary Computation 14(3):381–399.
[9] E. Rashedi, H. Nezamabadipour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm." Journal of Information of Science 179, 2232-2243, 2009.
[10] Rose Alqasem, Taisir Eldos, "An Efficient cell Placement Algorithm Using Gravitational Search Algorithms", Journal of Computer Science 9 (8): 943-948, 2013.
[11] Scott Kirkpatrick, "Optimization by simulated annealing: Quantitative studies", Journal of Statistical Physics, Vol 34, Issue 5-6, 975-986, 1984.
[12] Jin Xu, Albert Y.S. Lam, Victor O.K. Li, "Chemical Reaction Optimization for the Grid Scheduling Problem," IEEE Communications Society, publication in the IEEE ICC, 2010.
[13] Albert Y.S. Lam and Victor O.K. Li, "Chemical Reaction Optimization for Cognitive Radio Spectrum Allocation, "IEEE Communications Society, Proceedings of Globecom, 2010.
[14] A. Y. S. Lam, V. O. K. Li, and J. J. Q. Yu, "Real-Coded Chemical Reaction Optimization," IEEE Trans Evolutionary Computation, Vol. 16, No. 3, 339–353, Jun. 2012.
[15] K. Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal Operations Research 126, 106-130, 2000.
[16] Y. Chen, P. Zhang, Optimized Annealing of Traveling Salesman Problem from the nth-Nearest-Neighbor Distribution, Physica A: Statistical Mechanics and its Applications, Vol. 371, Issue 3, 627-632,2006.
[17] C.-N. Fiechter, A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems, Discrete Applied Mathematics 51, 243-26, 1994.
[18] M. Albayrak, N. Allahverdi, N.: Development a New Mutation Operator to Solve the Traveling Salesman Problem by Aid of Genetic Algorithms, Expert System Applications 38 , 1313-1320, 2011.
[19] N. Mladenović, P. Hansen, Variable neighborhood search, Computational Operations Research 24, 1097-1100, 1997.
[20] J.C. Créput, A. Koukam, A Memetic Neural Network for the Euclidean Traveling Salesman Problem. Neurocomputing 72,1250-1264, 2009.
[21] C.F. Tsai, C.W. Tsai, C.C. Tseng, A New Hybrid Heuristic Approach for Solving Large Traveling Salesman Problem, Information Science. 166,67-81, 2004.
[22] X.H. Shi, Y.C. Liang, H.P.Lee, C. Lu, Q.X. Wang, Particle Swarm Optimization-Based Algorithms for TSP and Generalized TSP, Information Proceeding Letter. 103,169-176, 2007.
[23] Helmi Md Rais, Zulaiha Ali Othman, Abdul Razak Hamdan, "Improved Dynamic Ant Colony System (DACS) on symmetric Traveling Salesman Problem," International Conference on Intelligent and Advanced Systems,43-48, 2007.
[24] Ahmed Rabie Ginidi Ginidi, Ahmed M. A. M. Kamel, Hassen Taher Dorrah, "Development of New Fuzzy Logic-based Ant Colony Optimization Algorithm for Combinatorial Problems, "Proceedings of the 14th International Middle East Power Systems Conference, Cairo University, Egypt, 2010.
[25] Wolpert DH, Macready WG (1997) No Free Lunch Theorems for Optimization. IEEE Trans Evolutionary Computation 1(1):67–82.